

test

SOME COMMANDS:

* ***fork***:

A copy of a repository that lives on your account. Forking a repository allows you to freely experiment with changes without affecting the original (upstream) project. Usually, forks are used to propose changes to a project or to use a project as a starting point for your own idea.

* ***clone****:*

A copy of a repository that lives on a local machine and can be edited offline.

*Ex:*

$ git clone <https address of repo>

* ***remote***:

The version of a repository hosted on a server. Use this to configure upstream repositories (i.e. the master origin on GitHub):

*Ex:*

View remote repositories that are configured:

$ git remote -v

Add an upstream repository:

$ git remote add upstream <https address of repo>

* ***branch:***

A parallel version of the master branch where you can freely work without disrupting the master. You can merge branches back to the master when you are ready to publish a change.

*Ex:*

View existing branches:

$ git branch

Make a new branch:

$ git branch <branch-name>

Move between branches:

$ git checkout <branch-name>

Delete branch (after a merge, ideally):

$ git branch -d <branch-name>

* ***add****:*

Incrementally add changes to the index for committing later. For example, you make changes to one code and start to edit another code, you might wanted to add these changes to the index first.

*Ex:*

Add all files to index:

$ git add -A

Add select files to index:

$ git add my\_code.py my\_code\_v2.py data.dat

Add using the wildcard character (\*) for every string available:

$ git add analysis\_1\* \*.py

* ***commit****:*

Update the changes made from the index or from working directory (local machine) to whatever branch you are currently on.

*Ex:*

Commit a change to the current branch you are on from the index:

$ git commit -m “give commit message (what was changed?)”

Commit all changes and bypass adding to the index:

$ git commit -a

* ***merge***:

Takes changes from one branch and applies them into another. This is done automatically when pulling from a remote repository or the origin.

*Ex:*

After making changes in some branch you want to merge into master:

$ git checkout master

$ git merge <branch-name>

Check the status:

$ git status

In the event there are unresolvable conflicts, edit the conflict files (indicated by *status*) in your favorite text editor or in vim:

$ vi conflict\_file.py

And find the <<<<<<< HEAD and >>>>>>> BRANCH-NAME tags and resolve the conflicts, save the file, and add to the index for committing

* ***pull****:*

Action of fetching in changes and merging.

*Ex:*

$ git checkout <branch-name>

$ git pull origin <branch-name>

* ***push***:

Send changes to remote repository.

*Ex:*

Push to remote master repository:

$ git push origin master

Push to remote branch:

$ git checkout <branch-name>

$ git push origin <branch-name>

NOTES:

* When editing files on your machine, remember that you are just not editing Git files of any branch or master, you are just editing local files in the working directory
* The Git files are in the ‘*.git’* directory and you must *commit* changes
  + You can stash changes for adding and committing at a later time:

$ git stash

$ git checkout <another-branch>

* Changes made to the master